Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 19, 2016

Y. Nishida
GE Global Research

October 17, 2015

# A-PAWS: Alternative Approach for PAWS
## draft-nishida-tcpm-apaws-02

## Abstract

This documents describe a technique called A-PAWS which can provide protection against old duplicates segments like PAWS. While PAWS requires TCP to set timestamp options in all segments in a TCP connection, A-PAWS supports the same feature without using timestamps. A-PAWS is designed to be used complementary with PAWS. TCP needs to use PAWS when it is necessary and activates A-PAWS only when it is safe to use. Without impairing the reliability and the robustness of TCP, A-PAWS can provide more option space to other TCP extensions.

## Status of This Memo

## Copyright Notice

Table of Contents

## 1. Introduction

PAWS (Protect Against Wrapped Sequences) defined in [RFC1323] is a technique that can identify old duplicate segments in a TCP connection. An old duplicate segment can be generated when it has been delayed by queueing, etc. If such a segment has the sequence number which falls within the receiver’s current window, the receiver will accept it without any warning or error. However, this segment can be a segment created by an old connection that has the same port and address pair, or a segments sent 2**32 bytes earlier on the same connection. Although this situation rarely happens, it impairs the reliability of TCP.

PAWS utilizes timestamp option in [RFC1323] to provide protection against this. It is assumed that every received TCP segment contains a timestamp. PAWS can identify old duplicate segments by comparing the timestamp in the received segments and the timestamps from other segments received recently. If both TCP endpoints agree to use PAWS, all segments belong to this connection should have timestamp. Since PAWS is the only standardized protection against old duplicate segments, it has been implemented and used in most TCP implementations. However, as some TCP extensions such as [RFC2018], [RFC5925] and [RFC6824] also requires a certain amount of option space in non-SYN segments, using 10-12 bytes length in option space for timestamp in all segments tends to be considered expensive in recent discussions.

In addition, although PAWS is necessary for connections which transmit more than 2**32 bytes, it is not very important for other connections since [RFC0793] already has protection against segments from old connections by using timers. Moreover, some research results indicates that most of TCP flows tend to transmit small amount of data, which means only small fraction of TCP connections really need PAWS [QIAN11]. Timestamp option is also used for RTTM (Round Trip Time Measurement) in [RFC1323]. Gathering many RTT samples from the timestamp in every TCP segment looks useful approach to improve RTO estimation. However, some research results shows the number of samples per RTT does not affect the effectiveness of the RTO [MALLMAN99]. Hence, we can think if PAWS is not used, sending a few timestamps per RTT will be sufficient.

Based on these observations, we propose a new technique called A-PAWS which can archive similar protection against old duplicates segments. The basic idea of A-PAWS is to attain the same protection against old all duplicate segments as PAWS while reducing the use of TS options in segments. A-PAWS is designed to be used complementary with PAWS. This means an implementation that supports A-PAWS is still required to supports PAWS. A-PAWS is activated only when it is safe to use. This sounds the applicability of A-PAWS is limited, however, we believe TCP will have a lot of chances to save the option space if it uses A-PAWS.

There are some discussions that PAWS can also be used to enhance security, however, we still believe that A-PAWS can maintain the same level of security as PAWS. Detailed discussions on this point are provided in Section 5. A-PAWS is an experimental idea yet, but we hope it will contribute to facilitating the use of TCP option space.

## 2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 3. The A-PAWS Design

A-PAWS assumes PAWS as it is designed to be used complementary with PAWS. Hence, a node which supports A-PAWS MUST support PAWS. The following mechanisms are required in TCP in order to perform A-PAWS.

## 3.1. Signaling Methods

An endpoint that supports A-PAWS can use the following signaling methods to activate A-PAWS logic.

1) Option Exchange in SYN
This method uses a new experimental TCP option defined in [RFC6994] and exchanges it during SYN negotiation. The format of the option is depicted in Figure 1. The option does not have any content as it simply indicates the endpoint supports A-PAWS. In this signaling method, when an endpoint wants to use A-PAWS, it MUST put A-PAWS option in SYN or SYN-ACK segment. If an endpoint does not find A-PAWS option in received SYN or SYN-ACK segment, it MUST not send segments with A-PAWS logic in Section 3.3. However, it MUST activate A-PAWS receiver logic in Section 3.4 if it has sent A-PAWS option in SYN or SYN-ACK segment. This is because some middleboxes may remove A-PAWS option in SYN or SYN-ACK segment. A-PAWS receiver logic in Section 3.4 can interact with both A-PAWS and PAWS sender. This signaling requires additional option space in SYN segments, hence non-SYN segment signaling should be used when there is not enough space in SYN option space.

2) Option Exchange in non-SYN Segments
This method uses the option in Figure 1 as well as the SYN segment signaling. However, the options are not exchanged during SYN negotiation. When a endpoint sets A-PAWS option in the segments, it indicates that it can receive the segments from A-PAWS senders. Hence, it MUST activate A-PAWS receiver logic in Section 3.4 if it sends the options. However, it MUST not send segments with A-PAWS logic in Section 3.3 until it receives A-PAWS options. This approach does not require extra option space or special timestamp value in SYN segments. However, negotiating features in non-SYN segments will require to address further arguments such as when to send the options or how to retransmits the options. We discuss these points in the next section and provide some recommended rules for implementations.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Kind = 254   |  Length = 4   |       16-bit ExID = TBD       |
+---------------+---------------+-------------------------------+
```

Figure 1: A-PAWS option format

### 3.2. A-PAWS Negotiation Logic for non-SYN Segment Signaling

One important characteristic for A-PAWS is its signaling mechanism does not require tight synchronization between endpoints since A-PAWS receivers can interact with both A-PAWS senders and PAWS senders. This allow us not to invent another three-way handshake like mechanisms for non-SYN segments. This approach will require drastic changes in the current TCP semantics. Instead, we propose a relatively simple and easy mechanism for feature negotiation by using the following rules on A-PAWS endpoints.

Rule 1: An endpoint MUST activate A-PAWS receiver logic in Section 3.4 before it sends A-PAWS option.

Rule 2: An endpoint MUST not send segments with A-PAWS logic in Section 3.3 until it receives A-PAWS option from the other endpoint.

These rules can avoid situations where an endpoint sends segments by A-PAWS logic to an endpoint that doesn’t use A-PAWS logic.

Another discussion point for this signaling method is when to set A-PAWS option in segments. As A-PAWS employs asynchronous signaling, both endpoints basically can set A-PAWS option in segments anytime they want. However, it is recommended to use the following rules for setting A-PAWS options.

Rule 3: An endpoint SHOULD use a data segment when it sets A-PAWS option in a segment.

Rule 4: When an endpoint receives a data segment with A-PAWS option, it SHOULD set A-PAWS option for its ACK segment.

Rule 5: An endpoint MAY use A-PAWS options in retransmitted segments.

These rules allow endpoints to have loose synchronized signaling so that they can at least solicit responses from their peers. Of course, even an endpoint solicit a response by setting A-PAWS option in a data segment, it might not receive A-PAWS option in the ACK segment. This can be caused by the lost of the ACK segment or middleboxes that remove unknown options. In order to address these cases, the following rules can be used.

Rule 6: As long as an endpoint does not violate the other rules, it MAY set A-PAWS option in multiple data segments with a certain interval in case no A-PAWS options has been sent from the peer.

This rule can address the cases where A-PAWS options has been removed by middleboxes or segments with A-PAWS options has been lost.

### 3.3. Sending Behavior

A-PAWS enabled TCP transmits segments, it needs to follow the rules below.

1. TCP needs to check how many bytes has been transmitted in a connection. If the transmitted bytes exceeds 2**32 - 'Sender.Offset', TCP migrates PAWS mode and MUST set timestamp option in all segments to be transmitted. The value for 'Sender.Offset' is discussed in Section 5.

2. If the number of bytes transmitted in a TCP connection does not exceeds 2**32 - 'Sender.Offset', TCP MAY omit timestamp option in segments as long as it does not affect RTTM. This draft does not define how much TCP can omit timestamps because it should be determined by RTTM.

### 3.4. Receiving Behavior

A-PAWS enabled TCP receives segments, it needs to follow the rules below.

1. TCP needs to check how many bytes has been received in a TCP connection. If it exceeds 2**32 bytes, A-PAWS nodes SHOULD discard the received segments which does not have timestamp option. TCP MUST perform PAWS check when received bytes exceeds 2**32 bytes.

2. If the number of bytes received in a TCP connection does not exceeds 2**32 bytes, A-PAWS nodes SHOULD accept the segments even if it does not have timestamp option. A-PAWS nodes MAY skip PAWS check until the received bytes exceeds 2**32 bytes.

## 4. When To Activate A-PAWS

In basic principal, A-PAWS capable nodes can always use A-PAWS logic as long as the peers agree with them. However, the following cases require special considerations to enable A-PAWS.

1. As "When To Keep Quiet" section in [RFC0793] suggests, it is recommended that TCP keeps quiet for a MSL upon starting up or recovering from a crash where memory of sequence numbers has been lost. However, if timestamps are being used and if the timestamp clock can be guaranteed to be increased monotonically, this quiet time may be unnecessary. Because TCP can identify the segments

from old connections by checking the timestamp. We think some TCP implementations may disable the quiet time because of using timestamps from this reason. However, since A-PAWS nodes does not set timestamp options in all segments, TCP cannot rely on this approach. To avoid decreasing the robustness of TCP connection, TCP MUST NOT use A-PAWS for a MSL upon starting up or recovering from a crash.

2. Various TCP implementations provide APIs such as setsockopt() that can set SO_REUSEADDR flag on TCP connections. If this flag is set, the TCP connection allows to reuse the same local port without waiting for 2 MSL period. While this option is useful when users want to relaunch applications immediately, it makes the TCP connection a little vulnerable as TCP stack might receive duplicate segments from earlier incarnations. It has been said that PAWS can contribute to mitigate this risk by checking the timestamps in segments. In order to keep the same level of protection, TCP SHOULD NOT send A-PAWS option when SO_REUSEADDR flag is set. This rule prevents the peer from sending segments to this node with A-PAWS logic. However, the node can send segments with A-PAWS logic as long as it received A-PAWS option from the peer.

## 5. Discussion

As A-PAWS is an experimental logic, the following points need to be considered and discussed.

### 5.1. Protection Against Early Incarnations

There are some discussions that timestamp can enhance the robustness against early incarnations. Since A-PAWS does not set timestamps in all segments, some may say that it degrades the robustness of TCP. We believe that the degradation caused by A-PAWS on this point is negligible. As long as TCP limits the usage of A-PAWS as described in Section 4, duplicate segments from early incarnations should not be received by TCP.

### 5.2. Protection Against Security Threats

A TCP connection can be identified by a 5-tuple: source address, destination address, source port number, destination port number and protocol. Crackers need to guess all these parameters when they try malicious attacks on the connection. PAWS can enhance the protection for this as it additionally requires timestamp checking. However, we think the effect of PAWS against malicious attacks is limited due to the simplicity of PAWS check. In PAWS, a segment can be considered as an old duplicate if the timestamp in the segment less than some timestamps recently received on the connection. The "less than" in this context is determined by processing timestamp values as 32 bit unsigned integers in a modular 32-bit space. For example, if t1 and t2 are timestamp values, t1 < t2 is verified when 0 < (t2 - t1) < 2**31 computed in unsigned 32-bit arithmetic. Hence, if crackers set a random value in the timestamp option, there will be 50% chance for them to trick PAWS check. Moreover, there will be more chances if they send multiple segments with different timestamps, which will not be difficult to perform.

In addition, we think there might be a case where using PAWS increases security risks. PAWS recommends to increase timestamp over a system when TCP waives the "quiet time" described in [RFC0793]. However, if timestamps are generated from a global counter, it may leak some information such as system uptime as discussed in [SILBERSACK05]. A-PAWS might be able to allows TCP to use random timestamp values per connections.

### 5.3. Middlebox Considerations

A-PAWS is designed to be robust against middleboxes. This means that endpoints will not be messed up even if middleboxes discard A-PAWS option. This is because A-PAWS sender logic is activated only when TCP receives a segment with A-PAWS options. A-PAWS receiver logic does not need to know whether the sender is using PAWS or A-PAWS. Activating A-PAWS receiving logic for PAWS sender might be redundant as it requires additional overheads. However, we believe the overhead will be acceptable in most cases because of the simplicity of A-PAWS logic.

Another concern on middleboxes is that they can insert or delete some bytes in TCP connections. If a middlebox inserts extra bytes into a TCP connections, there might be a situation where an A-PAWS sender can transmit segments without timestamp, while an A-PAWS receiver perform PAWS check on them as it already has received 2**32 bytes. In order to avoid discarding segments unnecessarily, we recommend that A-PAWS sender should have a certain amount of offset bytes in order to migrate PAWS mode before the receiver receives 2**32 bytes. We call this protocol parameter ’Sender.Offset’. The proper value for ’Sender.Offset’ needs to be discussed.

### 5.4. Aggressive Mode in A-PAWS

The current A-PAWS requires TCP to migrate PAWS mode after sending/ receiving 2**32 bytes. However, if both nodes check if 2 MSL has already passed during sending/receiving 2**32 bytes, it is safe to continue using A-PAWS. We call this Aggressive mode. The use of Aggressive mode will be explored in future versions.

## 6. Security Considerations

We believe A-PAWS can maintain the same level of security as PAWS does, but further discussions will be needed. Some security aspects of A-PAWS are discussed in Section 5.

## 7. IANA Considerations

This document uses the Experimental Option Experiment Identifier. An application for this codepoint in the IANA TCP Experimental Option ExID registry will be submitted.

## 8. References

### 8.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, DOI 10.17487/RFC1323, May 1992, <http://www.rfc-editor.org/info/rfc1323>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

### 8.2. Informative References

[MALLMAN99] Allman, M. and V. Paxson, "On Estimating End-to-End Network Path Properties", Proceedings of the ACM SIGCOMM , September 1999.

[QIAN11] Qian, L. and B. Carpenter, "A Flow-Based Performance Analysis of TCP and TCP Applications", 3rd International Conference on Computer and Network Technology (ICCNT 2011) , February 2011.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, DOI 10.17487/RFC2018, October 1996, <http://www.rfc-editor.org/info/rfc2018>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options", RFC 6994, August 2013.

[SILBERSACK05] Silbersack, M., "Improving TCP/IP security through randomization without sacrificing interoperability.", EuroBSDCon 2005 , November 2005.

## Author’s Address

Yoshifumi Nishida
GE Global Research
2623 Camino Ramon
San Ramon, CA 94583
USA

Email: nishida@wide.ad.jp